A Neural Network Based PID Controller for Underactuated Marine Craft Navigation

Austin Burch
Department of Mechanical Engineering
Texas A&M University
College Station, Texas 77840
Email: ajburch92@gmail.com

Mitchell Allain
Department of Mechanical Engineering
Texas A&M University
College Station, Texas 77840
Email: allain.mitch@gmail.com

***Abstract*—A gradient-based proportional-integral-derivative (GPID) tuning method and line-of-sight (LOS) heading algorithm have been developed for the heading control of a small under-actuated marine surface craft. The GPID algorithm utilizes a cumulative mean-squared-error cost function over a rotating memory and an approximate gradient descent to update the PID gain parameters in real-time. In addition, a nonlinear dynamic model is presented with its realization in MATLAB and Simulink. The gradient-based tuning algorithm is first evaluated for its properties over many simulations with randomly seeded initial conditions and environmental parameters. Then, the GPID controller is compared directly with a previously implemented proportional control algorithm, demonstrating reduced mean cross-track error under most trial conditions. We discuss the properties of the integrated navigational system as a whole and provide some practical considerations for future research.**

I. Introduction

The 'EMergency Integrated Lifesaving lanYard craft' (Hydronalix), or EMILY for short, is a mobile marine buoy platform which can be dispatched via remote control to a drowning victim by first responders. In many situations, teleoperation of such a vehicle is difficult for human operators due to a number of perceptual limitations and sensitive yaw dynamics. Therefore, the ability to autonomously transit to the victim along a prescribed path would provide a valuable tool for coast guard personnel in deploying multiple crafts over long distance rescue operations with limited manpower.

More precisely, the authors were tasked with improving the *mean cross-track error (MCTE)* of the existing proportional controller, using a machine learning-based control architecture. Cross-track error, $e_{\text{CTE}}[k]$, at each sample k is defined as the orthogonal distance to the desired path (straight or otherwise) from EMILY's current location, as illustrated in Fig. 1. The mean cross-track error, E_{MCTE}, over N samples is computed as

$$E_{\text{MCTE}} = \frac{1}{N}\sum_{k=1}^{N} e_{\text{CTE}}(k) \tag{1}$$

The existing navigational system is organized in the following manner: a UAV vision system, developed by Dufek, *et. al*, captures the craft's position and orientation, and transforms them to a global fixed frame [1]. Then, an LOS geometric function determines the best reference heading for path-keeping with a straight path, and a proportional controller

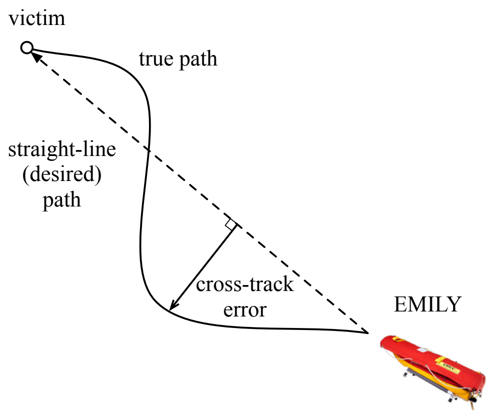


Fig. 1: Cross-track error

modifies the rudder position to achieve that heading reference, as detailed by Xiao, *et. al* [2].

Sec. II reviews the foundational literature on marine craft dynamics and control which are utilized in this project, as well as the relevant applications of learning-based controllers. Sec. III discusses modeling the kinematics and dynamics of EMILY in a maritime rescue environment. Sec. IV presents two line-of-sight geometric functions for determining the best heading reference for path-keeping, one of which is capable of supporting arbitrary parametric path functions, with some minor limitations.

Inspired by the PID neural network controller found in marine literature, the team implemented a gradient descent-based tuning method for a standard PID controller, as detailed in Sec. V. Two sets of experiments are conducted: one to study tuning properties of the algorithm, and one to benchmark performance against the existing proportional (P) controller. The experimental design is discussed in Sec. VI, with the results presented in Sec. VI-B. The final sections present some concluding remarks, recommendations for further investigation, and practical considerations.

II. Background and Literature Review

Marine craft control has a rich and illustrious history in the broader control systems domain. All marine craft, EMILY included, experience several low-level challenges including:

highly coupled and nonlinear dynamics, large degrees of freedom (DOF), and tendency of instability. In addition to these challenges, surface marine craft are often under-actuated and non-holonomic.

A proportional-integral-derivative (PID) controller is one of the oldest categories of linear feedback controller, and is used ubiquitously in industry. Incorporating an integral and derivative term allows the PID controller to react to error dynamics, typically improving tracking performance over the traditional P controller. In fact, the theoretical foundation for PID control was first developed by Minorsky for automatic ship steering systems [3]. For our purposes, employing an auto-tuned PID algorithm with resulting nonlinear dynamics rather than a nonlinear controller allows future collaborators to quickly understand the structure of the controller.

Following the marine path-keeping literature, we decompose the navigation problem into finding the path and then keeping the path. For the latter component, Fossen presented line-of-sight (LOS) guidance for marine craft, which provides a geometric and parametric approach to finding the reference heading towards a desired path [4]. By adjusting the LOS radius, the path-keeping aggressiveness can be tuned. However, this parameter does have a profound effect on the resulting controller dynamics.

Artificial neural networks (ANN) and the backpropagation algorithm provide a computational approach to learning complex functional relationships [5]. In the controls domain, several authors have begun utilizing ANNs to learn the black-box dynamics of a plant, or even in some situations, as the controller itself. In particular, Hernandez, *et. al* published a neural network tuning algorithm for control of underwater autonomous vehicles (UAVs), however, the authors use a sigmoid function on the output layer, which somewhat arbitrarily constrains the PID gains to the interval from zero to one [6]. Most neural network-based controllers use a tracking error metric as the loss function, and attempt to approximate the gradient with respect to the weights, as exact computation of the gradient requires a model of the plant.

Following this pattern, a simple framework is provided by Huailin [7], which employs P, I, and D type neurons, the latter two of which have first-order discrete dynamics, as a hidden layer in the network, and the reference value and measurement of the control signal as network inputs. Huailin also presents a method of approximating the gradient and backpropagating a cumulative tracking error loss function.

Nomenclature

Acronyms	
EMILY	EMergency Integrated Lifesaving lanYard
LOS	Line-of-Sight
Dynamics	
α_i	generic lumped parameter for control plant (EMILY)
θ	yaw (heading) angle
ζ	longitudinal displacement variable
u_1	Thrust control input, i.e., longitudinal velocity
u_2	Rudder angle control input, (ϕ)
Kinematics	
$\boldsymbol{\omega}_{nb}^{n}$	angular velocity vector of frame $\{b\}$ w.r.t frame $\{n\}$ in frame $\{n\}$
$\boldsymbol{p}_{b}^{n}$	vector to point b (or origin of frame $\{b\}$) in frame $\{n\}$
$\boldsymbol{p}_{ev}$	vector from EMILY to victim (global frame)
$\boldsymbol{v}_{nb}^{n}$	translational velocity vector of frame $\{b\}$ w.r.t point $\{n\}$ in frame $\{n\}$
e_θ	error between reference and measured heading
y_θ	measured value of heading ($= \theta$ above)
r_θ	LOS reference heading
R_{b}^{n}	rotates a coordinate vector in frame $\{b\}$ to frame $\{n\}$
Terms	
surge	longitudinal velocity in body frame
yaw	rotation about the azimuth axis
e_{CTE}	EMILY's cross-track error at particular sample
E_{MCTE}	EMILY's mean cross-track error

III. Dynamic Model and Simulation

A simplified parametric dynamic model was developed in order to prototype controllers. Without explicit formulation of the controller dynamics, this model allows designers to quickly test whether a particular learning algorithm is capable of controlling a plant with similar dynamics to the EMILY platform. Physical parameters are estimated where necessary. All of the necessary code to reproduce the simulations is available online[1].

A. Kinematics

A detailed review of marine craft kinematic modeling conventions can be found in the 2002 book by Fossen [8]. In our implementation, we adopt two conventional frames: the North-East-Down (global) coordinate frame and the body-fixed coordinate frame, which are denoted $\{n\}$ and $\{b\}$, respectively. A coordinate vector to a point a expressed in the frame $\{n\}$ is denoted $\boldsymbol{p}_{a}^{n}$. For example, $\boldsymbol{p}_{e}^{n}$ and $\boldsymbol{p}_{v}^{n}$ define the coordinate vector to EMILY and the victim in the inertial frame. These two frames are illustrated in Fig. 2.

The EMILY craft is non-holonomic, having only a rear rudder and longitudinal propulsion. For simplicity, only dynamics in the longitudinal (called surge) direction and yaw orientation are modeled here. *In the absence of current*, EMILY's velocity is denoted as

$$\boldsymbol{v}_{nb}^{b} = [\dot{\zeta}, 0, 0]^T \tag{2}$$

where $\boldsymbol{v}_{nb}^{b}$ is the velocity of the body frame with respect to the global frame, expressed in the body frame, and $\dot{\zeta}$ is

[1]https://github.com/MitchAllain/gpid-csce635

used to denote longitudinal velocity. The rotational velocity is expressed as

$$\boldsymbol{\omega}_{nb}^{b} = [0, 0, \dot{\theta}]^T \tag{3}$$

where $\boldsymbol{\omega}_{nb}^{b}$ is the angular velocity of the body frame with respect to the global frame, and θ denotes the yaw angle, or the angle between unit vectors n_1 and b_1. The rotation matrix R_b^n relates a coordinate vector in frame $\{b\}$ to frame $\{n\}$ and is defined as

$$R_b^n = \begin{bmatrix} n_1 \cdot b_1 & n_1 \cdot b_2 & n_1 \cdot b_3 \\ n_2 \cdot b_1 & n_2 \cdot b_2 & n_2 \cdot b_3 \\ n_3 \cdot b_1 & n_3 \cdot b_3 & n_3 \cdot b_3 \end{bmatrix}$$

$$= \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

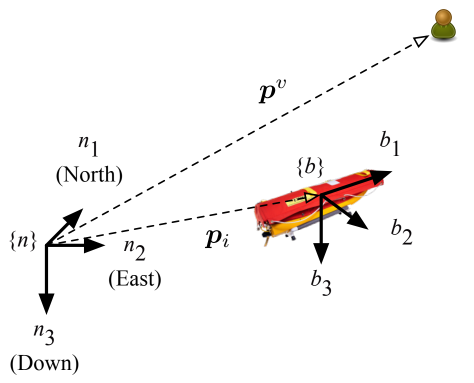


Fig. 2: Kinematic coordinate frames

In simulation, the effect of irrotational currents is considered as a constant additive velocity perturbation in the global frame, $\boldsymbol{v}_c^n = [c_x, c_y]$, yielding total translation velocity

$$\boldsymbol{v}_{nb}^{n} = R_b^n \boldsymbol{v}_{nb}^{b} + \boldsymbol{v}_c^n \tag{5}$$

Therefore, the position of EMILY at time t is given by

$$\boldsymbol{p}_e^n(t) = \int_0^t (R_b^n \boldsymbol{v}_{nb}^{b} + \boldsymbol{v}_c^n) d\tau + \boldsymbol{p}_e^n(0) \tag{6}$$

where $\boldsymbol{p}_e^n(0)$ is EMILY's initial position.

The orientation of EMILY at time t is given by

$$\Theta_{nb}(t) = \int_0^t \overset{I_{3\times3}}{\cancel{T_b(\Theta_{nb})}} \boldsymbol{\omega}_{nb}^{b} d\tau + \Theta_{nb}(0) \tag{7}$$

or simply,

$$\theta(t) = \int_0^t \dot{\theta} d\tau + \theta(0) \tag{8}$$

where $\theta(0)$ is EMILY's initial orientation in the global frame.

B. *Hydrodynamics*

The longitudinal equation of motion is given by:

$$\ddot{\zeta} = \frac{1}{m}[u_1 - \alpha_1 \dot{\zeta}^2] \tag{9}$$

where u_1 is the thrust force due to the rear jet and m is the mass of the craft. The lumped parameter α_1 describes hydrodynamic drag force, defined by

$$\alpha_1 = \frac{1}{2} C_D A_{fs} \rho \tag{10}$$

where C_D is the empirical drag coefficient and A_{fs} is the frontal submerged surface area of the craft.

Assuming irrotational currents, yaw rotation is caused by a turning moment produced by the rear rudder. The rotational dynamics can be described with the following equation of motion:

$$I\ddot{\theta} = \alpha_2 \cos(u_2) F_N - b\dot{\theta} \tag{11}$$

where I is the rotational inertia about the center of gravity, α_2 is distance from the craft's center of gravity (COG) to the rudder axis. F_N is the rudder drag force, and u_2 is the commanded angle for the rudder. Although the drag force does not act directly through the rudder axis, the distance has a minuscule effect on the resulting moment arm.

The drag force F_n on the rudder is determined semi-empirically from [9] using the following relation

$$F_N = \frac{1}{2} \frac{6.13\lambda}{\lambda + 2.25} A_r V_r^2 \rho \tag{12}$$

Here, ρ is the density of the fluid, λ is the aspect ratio of the rudder, A_r is the rudder area, and V_r is the relative velocity of the fluid, i.e.,

$$V_r = \dot{\zeta} - (\boldsymbol{v}_c \cdot \hat{b}_1) \tag{13}$$

where $(\boldsymbol{v}_c \cdot \hat{b}_1)$ is the magnitude of the current in the longitudinal direction of the craft.

Combined, Eqs. (3-5) become:

$$\ddot{\theta} = \alpha_2 \alpha_3 (\dot{\zeta} - (\boldsymbol{v}_c \cdot \hat{b}_1)^2 \cos(u_2) - \alpha_4 \dot{\theta} \tag{14}$$

with lumped parameters:

$$\alpha_3 = \left(\frac{1}{2I} \frac{6.13\lambda}{\lambda + 2.25} A_r \rho \right) \tag{15}$$

$$\alpha_4 = \frac{b}{I} \tag{16}$$

See the Sec IX-B in the appendix for the parameter values used in simulation and the listed state equations.

IV. Line-of-Sight Algorithms

Two line-of-sight (LOS) algorithms were developed simultaneously. The first is optimized for straight paths and utilizes the simplified geometry. The second LOS controller utilizes a multivariate unconstrained optimization approach to allow for generic reference paths, such as polynomial splines.

Note: all proceeding coordinate vectors are expressed in the global (fixed) frame, $\{n\}$, and the superscripts will be omitted for convenience.

A. Straight Path LOS

Given a straight reference path, the resulting LOS geometry is shown in Fig. 3, where the point i is EMILY's initial position, e is EMILY's current position, v is the victim, a is the orthogonal intersection with the straight path, and b is the LOS radius intersection with the path. The coordinate vector to point a is computed as

$$\boldsymbol{p}_a = \text{proj}_{\boldsymbol{p}_{ie}} \boldsymbol{p}_{iv} + \boldsymbol{p}_i = \frac{(\boldsymbol{p}_{ie} \cdot \boldsymbol{p}_{iv})}{||\boldsymbol{p}_{iv}||^2} \boldsymbol{p}_{iv} + \boldsymbol{p}_i \tag{17}$$

which is used to define the vector to the intersection point b

$$\boldsymbol{p}_b = \boldsymbol{p}_a + d_{ab} \frac{\boldsymbol{p}_{iv}}{||\boldsymbol{p}_{iv}||} \tag{18}$$

where d_{ab} is the distance between points a and b

$$d_{ab} = \sqrt{r^2 - e_{\text{CTE}}^2} \tag{19}$$

Here, r is the LOS radius parameter and e_{CTE} is the computed cross-track error

$$e_{\text{CTE}} = \frac{||\boldsymbol{p}_{ev} \times \boldsymbol{p}_{iv}||}{||\boldsymbol{p}_{iv}||} \tag{20}$$

The coordinate vector from EMILY to the intersection point is

$$\boldsymbol{p}_{eb} = \boldsymbol{p}_b - \boldsymbol{p}_e \tag{21}$$

where the heading reference angle can be computed

$$r_\theta = \text{atan2}(\boldsymbol{p}_{eb,y}, \boldsymbol{p}_{eb,x}) \tag{22}$$

Eqs. (17-22) are implemented in Algorithm 1.

Algorithm 1 Straight path LOS heading

```
1:  function STRAIGHT-LOS-HEADING($\boldsymbol{p}_e, \boldsymbol{p}_i, \boldsymbol{p}_v, r$)
2:      $\boldsymbol{p}_{ie} \leftarrow (\boldsymbol{p}_e - \boldsymbol{p}_i)$
3:      $\boldsymbol{p}_{iv} \leftarrow (\boldsymbol{p}_v - \boldsymbol{p}_i)$
4:      $\boldsymbol{p}_{ev} \leftarrow (\boldsymbol{p}_v - \boldsymbol{p}_e)$
5:      $e_{\text{CTE}} \leftarrow \frac{||\boldsymbol{p}_{ev} \times \boldsymbol{p}_{iv}||}{||\boldsymbol{p}_{iv}||}$
6:      $\boldsymbol{p}_b \leftarrow \text{proj}_{\boldsymbol{p}_{ie}} \boldsymbol{p}_{iv} + \boldsymbol{p}_i + \frac{\boldsymbol{p}_{iv}}{||\boldsymbol{p}_{iv}||} \sqrt{r^2 - e_{\text{CTE}}^2}$
7:      $\boldsymbol{p}_{eb} \leftarrow \boldsymbol{p}_b - \boldsymbol{p}_e$
8:      $r_\theta = \text{atan2}(\boldsymbol{p}_{eb,y}, \boldsymbol{p}_{eb,x})$    ▷ two argument arctangent
9:      return $r_\theta, e_{\text{CTE}}$
10: end function
```

B. Parametric Path LOS

The second LOS algorithm seeks to implement the same functionality with arbitrary parametric paths of type

$$\boldsymbol{f} : s \mapsto [x, y]^T \tag{23}$$

that is, the function maps a displacement parameter, s, to an (x, y) coordinate in the global frame. For instance cubic splines could be generated from a selection of waypoints [10].

The approach utilizes a parametric representation of the LOS circle around EMILY

$$\boldsymbol{p}_c(\psi) = \boldsymbol{p}_e + r \begin{bmatrix} \cos(\psi) \\ \sin(\psi) \end{bmatrix} \tag{24}$$

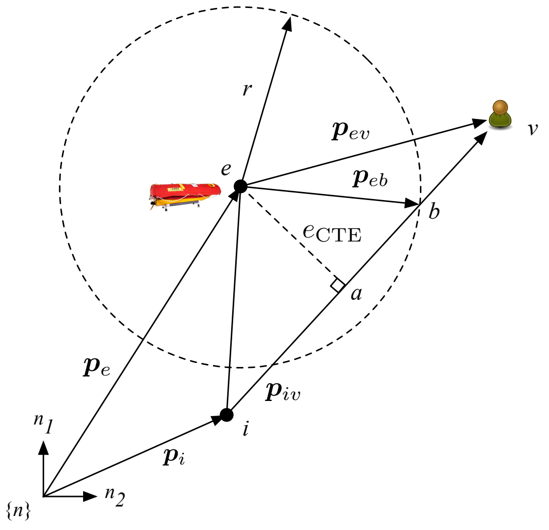


Fig. 3: Straight path LOS geometry

where $\boldsymbol{p}_c(\psi)$ is the coordinate of a point on the circle at angle ψ. Then, we define a cost function

$$J(\psi, s) = ||\boldsymbol{f}(s) - \boldsymbol{p}_c(\psi)||^2 \tag{25}$$

and run a multivariable minimization function such as MATLAB's `fminunc` (The MathWorks, Inc.), to result in the intersection parameters $[\psi^*, s^*]^T$, and intersection coordinates $\boldsymbol{p}_b = \boldsymbol{f}(s^*)$. Because there are typically two global minima (two intersections), the seeded parameter values for the search are the bearing to the victim and the point equal to the distance of one LOS radius projected straight forward from EMILY's position, and in the direction of EMILY's current heading. Still, the convergence properties of this approach are closely tied to the curvature of the path and the LOS radius parameter, and need to be studied more in depth.

V. Controller Design

As discussed previously, the PIDNN presents a method of tuning PID gains in real-time. The network, as presented by Huailin, is illustrated in Fig. 4a. The PIDNN network uses specialized P, I, and D type neurons with first-order discrete dynamics in the I and D neurons. The approach designates as a loss function the mean tracking error over a trailing window of samples, defined by

$$J = \frac{1}{m} \sum_{k=1}^{m} (r[k] - y[k])^2 \tag{26}$$

where m is the window (memory) length, and $r[k]$ and $y[k]$ are the reference and measurement.

Although this is not equivalent to mean cross-track error, minimizing the mean tracking error results in better path keeping and reduced cross-track error. Further, backpropagating the cross-track error directly would require knowledge of the second (thrust) input and violate our design constraint.

Algorithm 2 Arbitrary path LOS heading

```
 1: function F-PATH-LOS-HEADING($\boldsymbol{J}, \boldsymbol{p}_e, \boldsymbol{p}_i, \boldsymbol{p}_v, r$)
 2:     $\boldsymbol{p}_{ev} \leftarrow (\boldsymbol{p}_v - \boldsymbol{p}_e)$
 3:     $\boldsymbol{p}_{iv} \leftarrow (\boldsymbol{p}_v - \boldsymbol{p}_i)$
 4:     $s_0 \leftarrow \boldsymbol{f}^{-1}(\boldsymbol{p}_v)$                ▷ displacement at victim
 5:     $\psi_0 \leftarrow \text{atan2}(\boldsymbol{p}_{ev,y}, \boldsymbol{p}_{ev,x})$          ▷ bearing to victim
 6:     $[s^*, \psi^*]^T \leftarrow fminunc(J, [s_0, \psi_0])$
 7:     $\boldsymbol{p}_b \leftarrow \boldsymbol{f}(s^*)$
 8:     $\boldsymbol{p}_{eb} \leftarrow \boldsymbol{p}_b - \boldsymbol{p}_e$
 9:     $r_\theta = \text{atan2}(\boldsymbol{p}_{eb,y}, \boldsymbol{p}_{eb,x})$     ▷ two argument arctangent
10:     return $r_\theta$
11: end function
12:
13: function PARAMETRIC-CIRCLE, $\boldsymbol{p}_c(\psi, \boldsymbol{p}_e, r)$
14:     $\boldsymbol{p}_c \leftarrow \boldsymbol{p}_e + r \begin{bmatrix}\cos(\psi)\\ \sin(\psi)\end{bmatrix}$
15:     return $\boldsymbol{p}_c$
16: end function
17:
18: function COST, $J(\boldsymbol{f}, \boldsymbol{p}_c, s, \psi)$
19:     $J \leftarrow ||\boldsymbol{f}(s) - \boldsymbol{p}_c(\psi)||^2$
20:     return $J$
21: end function
```

Taking the gradient with respect to each of the weights from the hidden layer to the output layer requires that input to system gradient be approximated as

$$\frac{\partial y}{\partial u} \approx \frac{\Delta y}{\Delta u} = \frac{y[k] - y[k-1]}{u[k] - u[k-1]} \tag{27}$$

This approximation is the most significant limitation of the PIDNN approach, because it does not *gaurantee* that the weight updates will move towards a lower tracking error.

The authors suspect that the published success of this controller was in part due to seeding weights on the first layer of the network such that the input to the hidden layer was exactly

$$e = r - y \tag{28}$$

i.e., the input layer weights are seeded at 1 and -1. Further in any heading controller, all orientation values must be wrapped at the interval from $[-\pi, \pi]$, i.e., for state θ, each time step we enforce (using modular arithmetic)

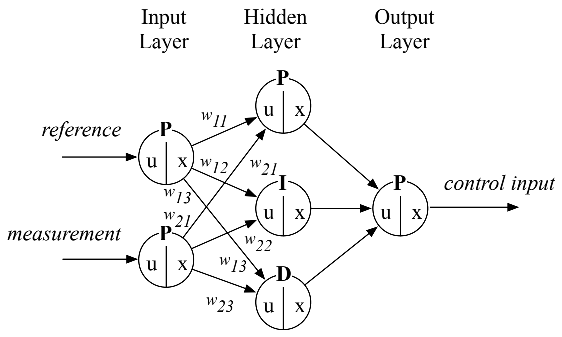
$$\theta_{\text{wrap}} = \theta + \pi \pmod{2\pi} \tag{29}$$

$$\theta = \theta_{\text{wrap}} - \pi \tag{30}$$

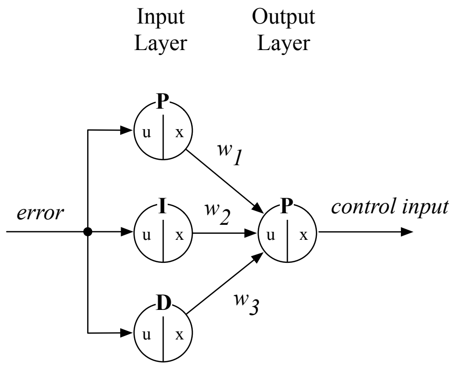
This modular heading computation is not possible within the original PIDNN framework. For these reasons, it is necessary to use a modified network architecture in our controller.

A. GPID Controller

By removing the input layer of the PIDNN, and replacing the inputs r_θ and y_θ with the error, e_θ, we arrive at the network shown in Fig. 4b, which is, admittedly, more appropriately designated as a *gradient-tuned PID (GPID)* on-line tuning algorithm. The loss function remains the same, as does the gradient with respect to the weights, which is



(a) PIDNN Controller [7]



(b) GPID "network"

Fig. 4: Comparison of PIDNN and GPID controllers

$$\frac{\partial J}{\partial w_j} \approx -\frac{2}{m}\sum_{k=1}^{m} e[k]\frac{y[k] - y[k-1]}{u[k] - u[k-1]}x_j[k] \tag{31}$$

The inside of the summation can be computed and stored at each time step to reduce computational cost, for which we will use the notation

$$\frac{\partial J}{\partial w_j} \approx -\frac{2}{m}\sum_{k=1}^{m} \delta_j[k]x_j[k] \tag{32}$$

The resulting in update rule can be written as

$$w_j[k+1] = w_j[k] - \alpha_j \frac{\partial J}{\partial w_j} \tag{33}$$

where α_j is an element of the learning rate vector. The pseudo-code for an update iteration of the GPID controller is shown in Algorithm 3. The resulting navigational system is shown in Fig. 5.

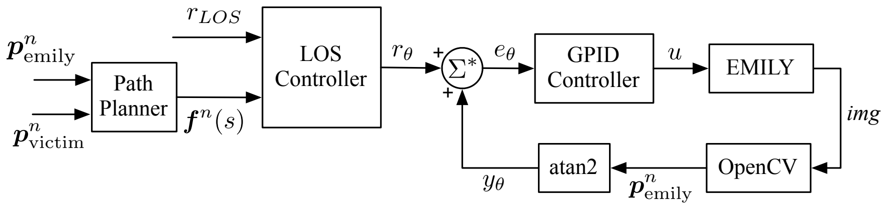


Fig. 5: The resulting navigational control system

Algorithm 3 Update the GPID controller

```
1:  function UPDATE-CONTROLLER($e_\theta, y_\theta, m$)
2:      $u, x_1, x_2, x_3, x_4 \leftarrow compute\_output(e_\theta)$
3:      push $y_\theta$ to measurement_mem                ▷ append to deque
4:      push $e_\theta$ to error_mem
5:      for each neuron $i$ do
6:          push $x_i$ to output_mem_i
7:      end for
8:      if size of error_mem $> 1$ then
9:          for each weight $j$ do
10:             $\delta_j \leftarrow$ Eq. 32                  ▷ incremental gradient
11:         end for
12:     end if
13:     if size of error_mem $== m$ then                  ▷ if deque is full
14:         for each weight $j$ do
15:             $\frac{\partial J}{\partial w_j} \leftarrow \frac{-2}{m} \sum_{k=1}^{m} \delta_j[k] x_j[k]$     ▷ sum deque
16:             $w_j \leftarrow w_j - \alpha_j \frac{\partial J}{\partial w_j}$                  ▷ backpropagate
17:         end for
18:         for each queue do
19:             pop oldest sample from queue
20:         end for
21:     end if
22:     return $u$
23: end function
```

VI. Experimental Design

The objectives of experimentation were to 1) test the convergence properties of the GPID gains and 2) verify that the GPID controller is superior to the previously implemented proportional controller. Further, we sought to document any practical limitations of the controller, and simulate the integration in the navigation system as a whole.

A. Simulation

All tests were executed using a combination of Python scripts, MATLAB functions, and Simulink block diagrams. Two Simulink models were developed interfacing the kinematic and dynamic model of EMILY with the LOS and controller functions: one for the GPID controller and one with a traditional proportional controller. The Simulink models were fully parametric, such that both the physical parameters of EMILY and the test environment parameters could be defined at execution. The simulations were terminated once EMILY reached within a 10 meter radius of the victim[2].

The output of each simulation includes two files, one with time series data for EMILY's position, measured heading, reference heading, the control input to the rudder, and the network weights if applicable, and one file with trial metadata such as path type, LOS radius, victim location, initial heading, ocean current velocity vector, throttle input value, network memory length, and network weight training rates.

The GPID controller itself was prototyped and tested in Python first for proof-of-concept. The calls to the network update function utilized MATLAB's ability to wrap Python objects and to convert the returned data to a MATLAB compatible data format.

B. GPID Tuning Properties

The GPID algorithm is parameterized by a memory length and a vector of learning rates, both of which have a significant effect on the controller dynamics. To test the properties of the GPID controller over repeated trials, several simulations were run with the parameters drawn uniformly from the intervals in Table I. EMILY began each trial at the origin of the global frame.

Parameter	Description	Lower	Upper
$p_{v,x}$	EMILY initial x	-50 m	50 m
$p_{v,y}$	EMILY initial y	-50 m	50 m
θ_i	EMILY initial heading	$-\pi$ rad	π rad
c_x	current in x direction	-0.66 m/s	0.66 m/s
c_y	current in y direction	-0.66 m/s	0.66 m/s

TABLE I: Intervals for randomly selected parameters

C. Performance Comparison

In order to compare the two controllers in a structured way, a set of trials were planned holding all parameters constant except for one. In this manner, the robustness of each controller to variations in specific parameters could be studied. The parameters are:

- the victim's position
- EMILY's initial heading
- the current direction and magnitude
- the throttle control input

[2]in implementation, the controller will switch to an entirely different scheme within a certain radius of the victim

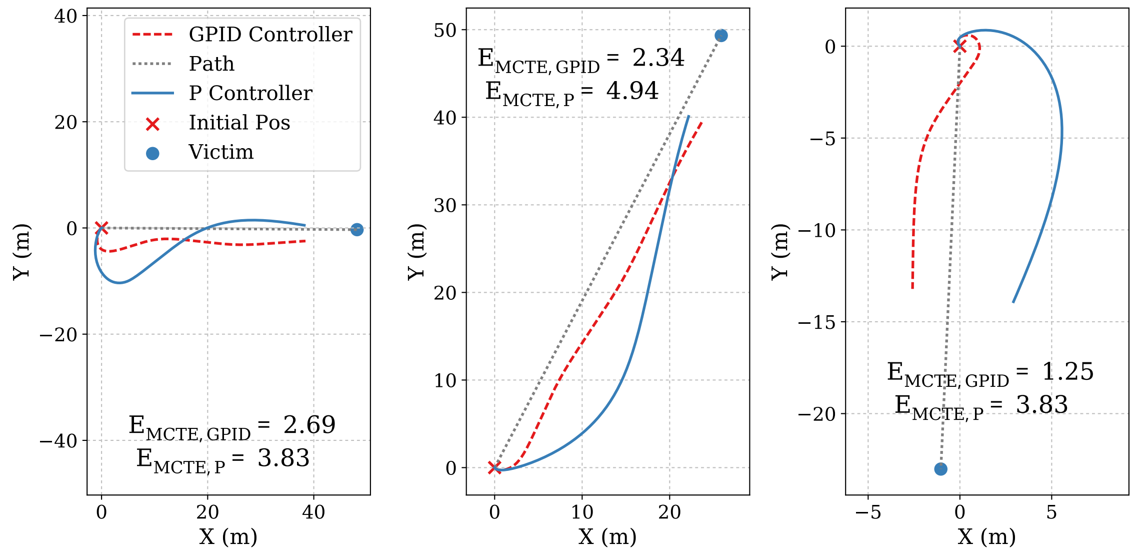


Fig. 6: Three sample simulation results with the proportional and GPID controller. *Note: the simulations terminate within a 10 m radius of the victim.*

- the path type

The values used in testing are displayed with the test results in the following section.

VII. Results and Discussion

A significant number of iterations were required to converge on acceptable parameters for the GPID algorithm. Further, the tuning action will not stabilize an unstable system, so the weights (gains) must be initialized with values from a stable controller.

Once an operating range had been established for the network parameters, the team ran batches of trials with parameters in the neighborhood to test the sensitivity of the algorithm. The results of changes on memory length and learning rate on weight convergence and cross-track error are illustrated in Figs. 7 and over the batch of trials. Since the cost function is constantly changing in response to the initial conditions and environmental dynamic of each trial, it is difficult to test for any properties of convergence.

However, a memory too short can lead to oversensitivity to the transient conditions like poor initial heading or environmental disturbances. This effect is evident in the rapid changes of gains in Fig. 7a. On the contrary, large memory inhibits the controller's ability to respond to transient conditions (note the larger distribution on the second controller in Fig. 8). A learning rate which is too aggressive can easily destabilize the system, however a learning too small tends to behave like a static PID system.

Sample trials for the GPID and proportional controllers are illustrated in Fig. 6, with mean cross-track error for both controllers inscribed in each subplot.

Structured trial parameters are displayed in Table II, along with the mean cross-track error for simulations with each controller. For each configuration (each row) all parameters were held constant *except* for the parameter in the left hand column. Notable scenarios where the controller performance differed greatly include traveling against the current, traveling *shorter* distances, and higher thrust values.

Param	Values	P MCTE	GPID MCTE
θ_i	180°	4.248	0.980
	135°	4.062	4.075
	225°	4.075	0.705
$\boldsymbol{p}_v$ (m)	$[25, 25]^T$	5.439	1.142
	$[40, 40]^T$	4.248	0.980
	$[100, 100]^T$	3.266	0.386
$\boldsymbol{v}_c$ (m/s)	$[0, 0]^T$	4.248	0.980
	$[0.3, 0.3]^T$	10.405	7.426
	$[-0.3, -0.3]^T$	3.616	0.366
	$[0.6, 0.6]^T$	5.955	1.507
	$[-0.6, 0.6]^T$	45.437	39.405
	$[-0.6, -0.6]^T$	8.055	0.524
Path Type	*straight*	4.248	0.980
	cubic 1	4.371	1.998
	cubic 2	4.446	3.463
	cubic 3	4.423	2.322
thrust (N)	1.5	4.944	1.144
	3	4.248	0.980
	6	3.573	0.409
	Average	**7.754**	**4.162**

TABLE II: Varied parameters and effect on MCTE

VIII. Conclusion

The authors were tasked with improving and benchmarking a navigational control system for a mobile marine buoy. Two line-of-sight geometric algorithms were presented which produce reference heading angles that converge to the desired path, either expressed as a straight line or a parametric curve. Then, a gradient-based proportional-integral-derivative (GPID) tuning method is discussed which utilizes a cumulative mean-squared-error cost function over a rotating memory and an approximate gradient descent to update the PID gain parameters in real-time.

The results from all simulations lead our team to believe the GPID approach to tuning PID control parameters is significantly superior to the existing proportional controller. Algorithms for GPID and LOS control consistently produced lower cross-track error. The increased stability due to the GPID controller leads to shorter arrival times and more robustness to external perturbations. The gradient descent approximation gives EMILY the functionality to adapt to varying environments, form factors, and weight distributions.

Acknowledgment

The authors would like to thank Dr. Murphy and her team of research assistants. A special thanks to Jan Dufek, who provided both insight and expertise throughout the development of this project.

IX. Appendix

A. Recommendations

The authors are confident that a PID-based control method is adequate for this system, although they suspect that the GPID controller, at its current level of development, may be too brittle for deployment. At a minimum, the weights should be saturated on an interval of known stable controllers. That way, a failed trial or the presence of a large disturbance could not completely disrupt the learning process, causing instability, rather the system would just degrade to a suboptimal, but stable, PID controller.

B. State Equations

The states variables and state equations are presented below in Table III and Eqs (34-37).

$$\dot{x}_1 = x_2 \tag{34}$$

$$\dot{x}_2 = \frac{1}{m}[u_1 - \alpha_1 x_2^2] \tag{35}$$

$$\dot{x}_3 = x_4 \tag{36}$$

$$\dot{x}_4 = \alpha_2\alpha_3(x_2 - (\boldsymbol{v}_c \cdot \hat{b}_1))^2 \cos(u_2) - \alpha_4 x_4 \tag{37}$$

State	Description
x_1	Longitudinal displacement in body frame, ζ
x_2	Longitudinal velocity in body frame, $\dot{\zeta}$
x_3	Yaw angle, θ
x_4	Yaw velocity, $\dot{\theta}$

TABLE III: Dynamic model states

Parameter	Value	Units
b	10	Ns/m
m	40	kg
C_D	0.6	$-$
A_{fs}	0.03	m^2
ρ	1027	kg/m^3
λ	1.6	$-$
A_r	0.015	m^2
I	7.5	kg/m^2

Lumped Parameter	Value	Units
α_1	9.2	kg/m
α_2	0.5	m
α_3	2.31	$1/m^3$
α_4	1.33	$1/s$

TABLE IV: Simulation parameters, approximated for EMILY

References

[1] J. Dufek and R. Murphy, "Visual pose estimation of usv from uav to assist drowning victims recovery," in *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 147–153.

[2] X. Xiao, J. Dufek, T. Woodbury, and R. Murphy, "Uav assisted usv visual navigation for marine mass casualty incident response," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, 2017.

[3] N. Minorsky, "Directional stability of automatically steered bodies," *J. Amer. Soc. Navel Engineers*, vol. 42, no. 2, pp. 280–309, 1922.

[4] T. I. Fossen, M. Breivik, and R. Skjetne, "Line-of-sight path following of underactuated marine craft," *Proceedings of the 6th IFAC MCMC, Girona, Spain*, pp. 244–249, 2003.

[5] R. Rojas, *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

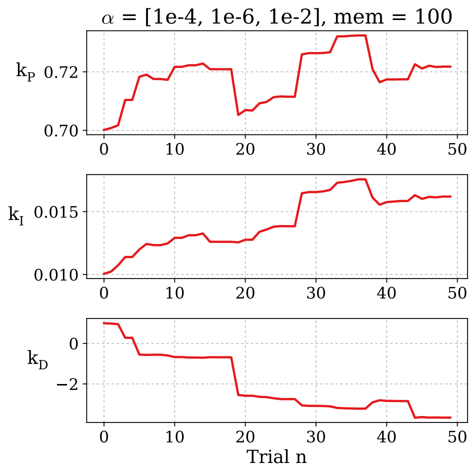
[6] R. Hernández-Alvarado, L. G. García-Valdovinos, T. Salgado-Jiménez, A. Gómez-Espinosa, and F. Fonseca-Navarro, "Neural network-based self-tuning pid control for underwater vehicles," *Sensors*, vol. 16, no. 9, p. 1429, 2016.

[7] H. Shu and Y. Pi, "Pid neural networks for time-delay systems," *Computers & Chemical Engineering*, vol. 24, no. 2, pp. 859–862, 2000.

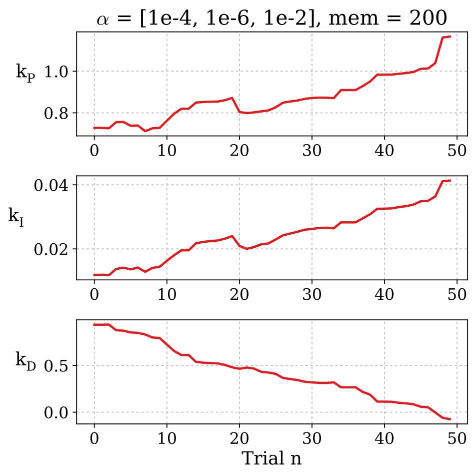
[8] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[9] K. Żelazny, "Approximate method of calculation of the wind action on a bulk carrier," *Zeszyty Naukowe/Akademia Morska w Szczecinie*, 2014.

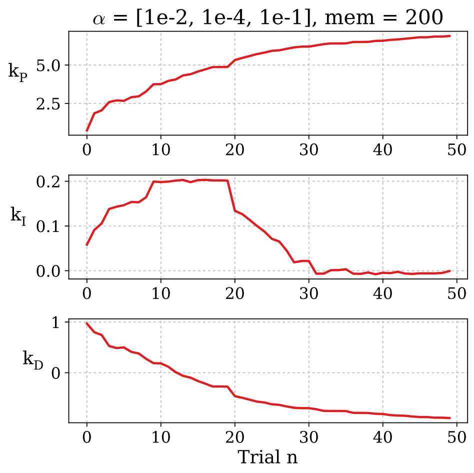
[10] C. De Boor, C. De Boor, E.-U. Mathématicien, C. De Boor, and C. De Boor, "A practical guide to splines," 1978.



(a) Short memory and low learning rate



(b) Long memory and low learning rate



(c) Long memory and high learning rate

Fig. 7: Effect of learning rate and memory on gain convergence

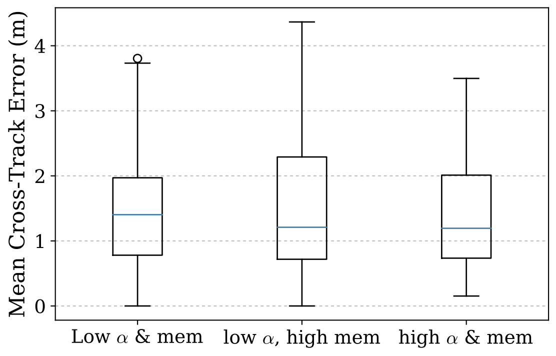


Fig. 8: Mean cross-track error for the three GPID parameter sets, i.e., training rate vector α and memory, m